Ising模型的并行计算*

刘军 沈扬 罗向前[1)]
(中山大学物理系 广州 510275)

摘要 以Ising模型为例,介绍有关格点系统的Monte Carlo数值模拟并行算法的设计和编程,并给出在本组建造的PC集群式高性能并行计算系统上的测量结果.本文的结果对格点量子色动力学的大规模数值模拟研究有一定的参考价值.

关键词 蒙特卡罗模拟 Ising模型 格点规范理论 并行算法 高性能并行计算系统

1 引言

最近获批准的国家自然科学基金重点项目"格点规范理论的大规模数值模拟研究"[1],成员由中山大学、北京大学、浙江大学和中科院高能物理研究所等单位组成.目的是利用国内日益发展的高性能计算系统,从第一原理出发,探索计算非微扰物理量的新方法,对胶球和混杂态等新强子态性质、强子散射过程、规范场真空拓扑性质、夸克禁闭机制、夸克胶子等离子体相变、量子瞬子、量子混沌等热门课题开展格点规范理论的数值模拟研究.本项目的主要实验工具是大规模计算机模拟.单靠一个或多个电脑独立工作,无法处理更大体积的格点,需要用到高性能计算机和大量计算机时,而并行计算能较好地解决这些问题.

本文以二维Ising模型为例,介绍数值模拟及其并行化程序的分析、设计和实现,同时简单介绍格点量子色动力学的基本思想和在本组的高性能并行计算系统一些测试结果.

2 Ising模型和Monte Carlo模拟

Ising模型[2]是在统计物理中最简单的格点系统之一.二维Ising模型有精确解,可以用此来作为基准检验其他近似方法的可靠性.并着重讨论如何用Monte Carlo模拟方法来研究并如何提高其计算效率.要得到更加精确的结果需要很大数目的格点,这就需要并行计算机.

二维的Ising模型可描述为:设有 $V = L \times L$ 个自旋,处于晶格格点位置,每个自旋只能取向上或向下两个态,如图1所示,并只考虑近邻自旋之间的相互作用,其哈密顿量为

$$H = -J\sum_{i<j} S_i S_j - \mu B \sum_{i=1}^{V} S_i, \qquad (1)$$

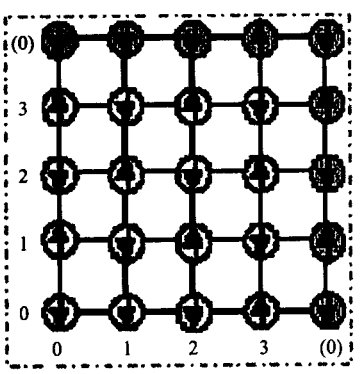

图1 二维4×4的格点和自旋组态

其中 S_i 代表第 i 个格点位置的自旋,取值为+1或-1,分别对应于自旋向上或向下.J 为耦合常数.这里令 $J>0$,H 描述铁磁体(因为 $B=0$ 时,其基态是全部自旋取向相同的态).若 $J<0$,则描述反铁磁体.式中第二项代表在外磁场 B 中的塞曼能,μ

2003-05-26收稿,2003-07-27收修改稿
* 国家自然科学基金(10235040),广东省教育厅,中山大学高等学术中心基金资助
1)通讯联系人,E-mail:stslxq@zsu.edu.cn

为与自旋相应的磁矩. 真正的物理系统对应于热力学极限 $V\to\infty$ 的情形,但实际计算只能对有限的 V 进行. 为了考虑 H 中的第一项在边界的效应,一般采用周期边界条件.

统计物理通常要计算一些物理量的统计平均值,如系统内能和比热:

$$U = \langle H\rangle,$$
$$C = \frac{\partial U}{\partial T}, \tag{2}$$

其中某一物理量 A 的统计平均值为$\langle A\rangle$定义为

$$\langle A\rangle = \frac{\sum_{\{S\}} A(\{S\})e^{-\beta H(\{S\})}}{\sum_{\{S\}} e^{-\beta H(\{S\})}}. \tag{3}$$

以上公式中,分母是配分函数,T 是温度,$\beta = 1/(k_B T)$,k_B 是 Boltzmann 常数,而求和对所有可能的自旋组态进行.

对于 V 个格点的系统,这样的求和的项就有 2^V,即随着格点的大小指数地增长,要产生所有这些组态可能远超出所有最先进的计算设备的能力. 如果用计算机求解,最有效是采用 Monte Carlo 重点抽样法来计算. 即用计算机产生统计独立的、有代表性的、对求平均贡献最大的组态,满足 Boltzmann 分布

$$P(\{S\}) = \frac{e^{-\beta H(\{S\})}}{\sum_{\{S\}} e^{-\beta H(\{S\})}}. \tag{4}$$

从平衡态开始,抽出其中 N 个组态并测量有关物理量. 这样,A 的统计平均值可近似为

$$\langle A\rangle = \frac{1}{N}\sum_{n=1}^{N} A(\{S\}_n). \tag{5}$$

Metropolis 抽样方法[3]是实现产生组态的重点抽样法之一,除此之外,还有热浴法(Heatbath)和杂化(Hybrid) Monte Carlo 法等. Metropolis 算法的基本思想是,通过一个适当的跃迁概率 $W(\{S\}_n\to\{S\}_{n+1})$:

$$W(\{S\}_n\to\{S\}_{n+1}) = \min\left[1, \frac{P(\{S\}_{n+1})}{P(\{S\}_n)}\right], \tag{6}$$

其中 $\delta H = H(\{S\}_{n+1}) - H(\{S\}_n)$. 可以证明,按上式满足细致平衡条件. 以上过程的伪码见附录 A.

3　格点规范理论

粒子物理和规范场论是探索最深层次物质世界的前沿科学,量子色动力学(QCD)是描述物质结构最基本的单元、即夸克与胶子间强作用的规范理论. QCD 的特点是低能区相互作用非常强,微扰论完全失效,只能用非微扰方法处理. 由诺贝尔奖得主 Wilson 所建立的格点规范理论,是处理强作用非微扰的最可靠工具. 它还与计算科学紧密结合起来,推动物理学和其他学科的发展. 格点规范理论的基本思想是:把四维连续时空用离散的晶格代替,这样,连续理论无法计算(发散的)物理量就可在格点上作 Monte Carlo 数值模拟计算,最后还要用重整化群方法把结果外推到连续极限. 当然这个工作量是非常巨大的,要用到高速并行计算机. 以纯规范场为例,把规范场定义连接格点间的链上

$$U(x,\mu) = \exp\left[-ig\int_{x'a}^{x'a+a\hat{\mu}} dx' A_\mu(x')\right]. \tag{7}$$

格点之间的距离是 a(晶格常数). 格点位置为 x,μ 为链的方向. 其 Wilson 作用量为[4]

$$S_g = -\beta\sum_{x,\mu<\nu} P_{\mu\nu}, \tag{8}$$

其中 $P_{\mu\nu} = \mathrm{Tr}(U_p + \mathrm{h.c.})/6$,$U_p$ 绕在格点位置为 x,方向为 μ 和 ν 的 1×1 方块的 4 条规范场链的乘积. $\beta = 1/g^2$,而 g 是规范场的耦合常数. 可以证明,在连续极限 $a\to 0$,以上作用量回复到 Yang-Mills 作用量.

规范场的配分函数是

$$Z = \int [dU]\exp(-S_g). \tag{9}$$

像 Ising 模型那样,可以用重点抽样法(较常用的是热浴法和杂化 Monte Carlo 法)来产生满足 Boltzmann 分布 $\exp(-S_g)/Z$ 的规范链组态,然后对各种物理量进行测量.

当然,格点 QCD 的方法有系统的误差. 众所周知,Wilson 规范场作用量与 Yang-Mills 作用量之间有 $O(a^2)$的差别. 只有 a 非常小时,这个误差才会变小. 为了消除有限体积效应,需要很大的格点体积 V,这无疑会耗费更多的计算资源. Symanzik 改进方案[5]的想法是:通过把次近邻的项加入 Wilson 作用量来减小格距误差,例如,一个最简单的改进作用量为

$$S_g = -\beta\sum_{x,\mu<\nu}\left(\frac{5}{3}P_{\mu,\nu} - \frac{1}{12}R_{\mu,\nu}\right), \tag{10}$$

其中 $R_{\mu\nu}$ 对应于 2×1 和 1×2 的长方形的 6 条规范场链的乘积的项.

与 Tadpole 改进[6]或非微扰改进[7]结合起来,Symanzik 改进 QCD 方案已导致十分有意义的进展,使在粗糙的格点上获得趋近物理连续极限的物理结果成为可能.

4 并行算法的设计

并行算法是指一个过程可分配到不同的子进程在不同的处理器上同时处理的计算方法. 以上格点系统十分容易并行化:即可把大格点划分成几块,分别分配给不同的子进程执行组态更新,最后再把更新了的数据归纳于一起进行测量. 现在可以初步预见,对于较大的格点,如果通信/计算时间比是较小的,则可使用并行计算将大大提高计算时间.

主体算法描述如图2. 以二维 Ising 模型为例,根据其特点,可采用以下并行算法. 先在主任务中把二维格点划分成数条[2],然后分配到各个处理器或进程中去,每个处理器或进程再在自己的内存空间中执行分配给它的子任务,每一子任务代表完成对某一个分条块作一次子组态的更新,并且只测量此条格点的数据,如磁场强度和内能. 最后由 MPI 函数 MPI-ALLReduce 将所有的处理器或进程中的数据归纳,得到整个格点的总体数据.

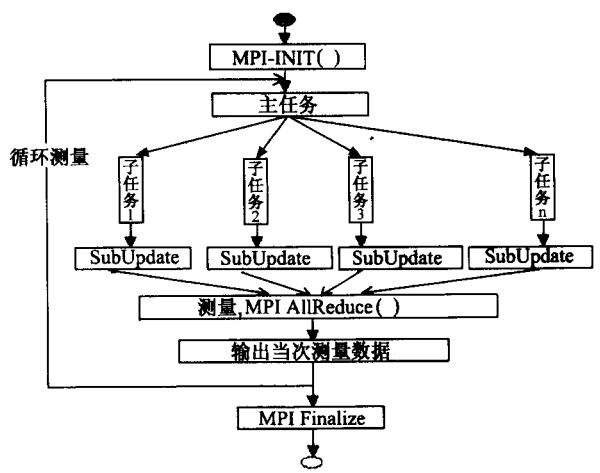

图2 并行算法主要流程

由于并行程序的特殊性,其算法设计远不同于串行程序,主要难点在于任务划分,进程间通信,此外还涉及网络拓扑、采用的语言及支持库等. 这里采用了 MPICH 作为并行开发环境. 以下来看几个技术要点.

条分解(Strip Decomposition)是把主任务分配给子任务的方法[8]. 假设有 P 个处理器(分别标为0,1,2…,$P-1$),就可把大格点分成 L/P 个子格点,如图3所示,其中每块大小为 $L/P \times L$,分配给处理器 P_α 的处理单元为从第 $(L/P)\times\alpha$ 行到 $[(L/P)\times(\alpha+1)]-1$ 行. 图中 $L=16, P=4, \alpha=0,1,2,3$.

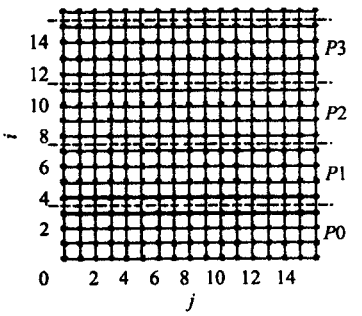

图3 条分解

(1)式中的 H 描述整个格点上所有相邻自旋的相互作用,要把 H 对所有子格点求和. 在划分块之后,每块子格点的头尾两行要从其他进程(处理器)中的相应行取得数据,要通过进程间通信来实现. 此处采用进程间缓存(Interprocessor Caching)的技术来解决. 在子格点的头尾两行上再加两行缓存行(Caching Row),用来接收和发送数据给近邻的进程,如图4所示. 为了确切自身的进程号及其上下相邻的进程号,每个进程需要设置以下变量:

int pid—自身的进程号;

int up_id = (pid + 1)% P—处理上一块子格点的进程号;

int lw_id = (pid + P - 1)% P—处理下一块子格点的进程号.

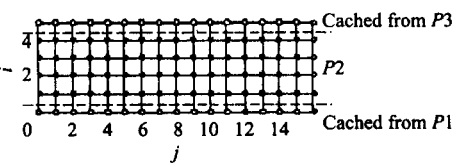

图4 缓存模型

因此,设置子格点的自旋数组大小为 $S[(L/P)+2][L]$,多出来的两行用来放置缓存行.

5 编程

对于二维 Ising 模型,用 C 语言编程,并用 MPI 把程序并行化. 以下给出程序的主要思路,具体源码在附录 B,C,D,E 给出.

程序由主函数 main,子函数 simulate, update 组成. 其中主进程主要执行任务分发,最后将所有子进程的数据集中起来进行下一步的数量分析. 各子进程在接受到所分配的子任务后将读取各自所需的数据(readData 函数),然后初始化各自的格点自旋初态,接着开始调用 simulate 子函数实行模拟过程.

其中涉及到对自旋组态的多次更新,这里通过多次调用 update 子函数实现对格点自旋组态的更新. Update 子函数中调用 row-update 来实现 Metropolis 算法. 由于在扫描格点的首尾两行时需要相邻进程的数据,所以这里需要用到上面所说的"进程间缓存"技术来实现,具体算法可以参见附录.

各进程完成各自的数据处理后就由请进程将数据归约起来进行数据分析,由主进程调用 measure 子函数来实现.

6　结果分析

2000 年初,本组建造了一套有 20 个 CPU 的 PC 集群式高性能并行计算系统[9—12]. 其基本配置为

(1) 硬件配置:

10 个节点,每个节点有双 CPU PIII 500MHz;

每个节点内存:128MB;

交换机速度:100MBbit/s,

(2) 软件环境:

操作系统:RedHat Linux;

并行计算环境:MPICH.

对于 $k_B = J = 1, B = 0$ 的二维 Ising 模型,测量了系统的平均内能和平均比热等物理量. 结果如图 5 所示. 从结果同准确解比较可以看出,计算机模拟和准确解符合较好. (测量次数 1000,组态间隔 200,格点大小 60×60).

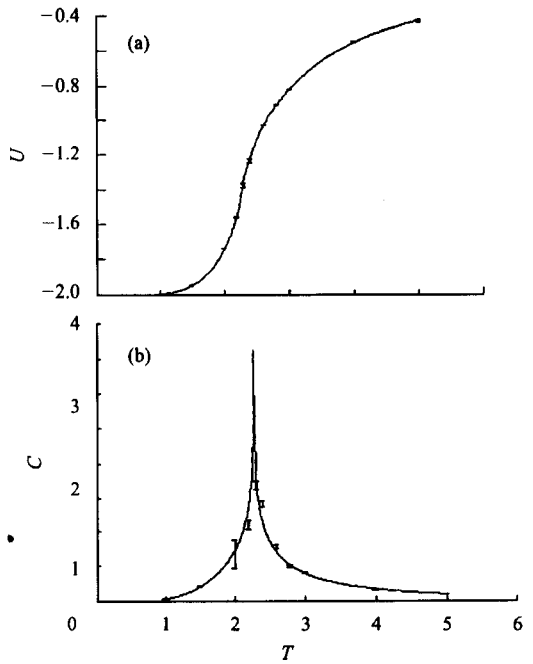

图 5

(a) 内能随温度的变化; (b) 比热随温度的变化

图 6 显示二维 Ising 模型计算机时与格点大小、所用 CPU 数目的关系. 由图可以看出,在格点总数 $V = L \times L$ 不大的情况下,并行计算的优势并不能很好地体现出来. 如 $L = 20$,40 和 60 的情况下,用两个 CPU 计算只比用一个 CPU 计算的时间略快,加速比在 1.3—1.9 附近,当增加 CPU 数时,计算时间并没有多大的缩短,甚至有增长的情况出现(如 $L = 20$ 时). 这是由于 L 不大的时候,每个 CPU 的计算量较少,通信时间占了总的计算时间的较大比例,所以加速比不好. 当体积增大到一定程度,例如上图中的 $L \geqslant 100$ 时,并行计算的优势就能较好地体现出来. 如用两个 CPU 的加速比很接近于 2. 用多个 CPU 计算时,时间也有较大的减少. 这是因为当 CPU 间的通信时间比每个 CPU 的计算时间小得多时,并行计算的优势才能很好地体现出来加速比也会上升.

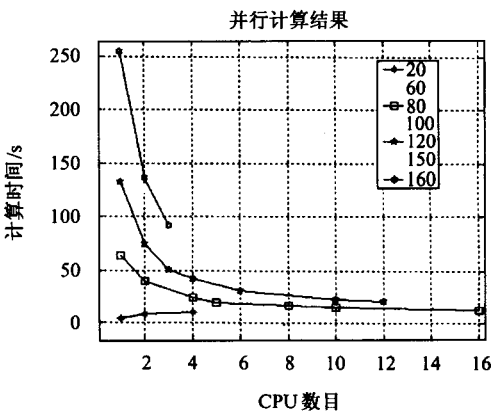

图 6　Ising 模型计算时间与 CPU 数目的关系

以上 Ising 模型的计算方法和结果对格点规范理论的大规模数值模拟研究很有启发作用. 格点 QCD 的并行算法也是把大格点分成多个子格点在不同的 CPU 运算. 在子格点的边界,必然有信息交换. 幸好,这种信息交换量并不大. 在本组的并行计算系统上,用(10)式给出的最简单的改进纯规范场作用量进行Monte Carlo模拟,格点体积 $V = 10^4$. 测

表 1　本组计算系统的改进格点 QCD 的运行结果,并与其他计算系统比较

机器	更新每条链所花机时: μs/链	节点间的通信速度: Mbits/s
SX-4	4.50	45
SR2201	31.4	28
Cenju-3	57.42	8.1
Paragon	149	9.0
本组并行计算系统	4.0	10.35

试了更新每条链所花的计算机时和节点间的通信速度.表1给出本组的测试数据,并与其他系统比较[13].还在本组的系统中,运行MILC的格点QCD并行计算程序库[14],结果是令人鼓舞的[15].

我们已经利用本系统,用格点QCD计算了轻强子[16]、胶球[17]和混杂态[18,19]的质量,这些结果和经验,对利用国内更高性能并行计算系统,进行更大规模的格点规范理论数值模拟研究,相信是十分有意义的.

国内其他组对格点QCD的研究,综述见文献[20].胶球新算符的构造和相应的质量谱计算见文献[21].

感谢Eric B. Gregory的讨论和帮助.

参考文献(References)

1 http://qomolangma.zsu.edu.cn/KEY_PROJ/index.html

2 Ising E. Z. Phys., 1925, **31**:253—258

3 Metropolis N et al. J. Chem. Phys., 1953, **21**:1087—1092

4 Wilson K. Phys. Rev., 1974, **D10**:2445—2459

5 Symanzik K. Nucl. Phys., 1983, **B226**:187—204

6 Lepage P, Mackenzie P. Phys. Rev., 1993, **D48**:2250—2264

7 Luscher M. hep-lat/9802029

8 Nakano A. High Performance Computing, http://www.cclms.lsu.edu

9 LUO X Q et al. In: Nonperturbative Methods and Lattice QCD, X.Q. Luo, E. Gregory eds. Singapore: World Scientific, 2001, 223

10 LUO X Q et al. In: Advanced Computing and Analysis Techniques in Physics Research, American Institute of Physics, 2001, 270[CS.DC 0109004]

11 LUO X Q et al. Nucl. Phys., 2002, **B106**(Proc. Suppl.):1046—1048

12 LUO X Q et al. Parallel Computing for QCD on a Pentium Cluster. hep-lat/0011090

13 Hioki S, Nakamura A. Nucl. Phys., 1999, **B73**(Proc. Suppl.):895—897

14 http://www.physics.indiana.edu/~sg/milc/benchmark.html

15 LUO X Q et al. Commun. Theor. Phys., 2003, **40**:319—328

16 MEI Z, LUO X Q, Gregory E. Chin. Phys. Lett., 2002, **19**:636—638

17 MEI Z, LUO X Q. National Conference on High Energy Physics, 2002

18 LUO X Q, MEI Z. Nucl. Phys., 2003, **B119**(Proc. Suppl.):263—265

19 MEI Z, LUO X Q. to appear in Int. J. Mod. Phys. A. [hep-lat/0206012]

20 LUO X Q. HEP & NP, 1999, **23**:188—194(in Chinese)
(罗向前.高能物理与核物理, 1999, **23**:188—194)

21 LIU D Q, WU J M. HEP & NP, 2002, **26**:222—229(in Chinese)
(刘大庆,吴济民.高能物理与核物理, 2002, **26**:222—229)

附录A Metropolis抽样方法的伪码

for($i=1$; $i<V$; i ++){随机从格点中选一格点,并设当前系统状态为$\{S\}_n$;

求此格点四周格点的自旋值;

定义此格点自旋翻转后的系统的状态为$\{S\}_{n+1}$;

计算翻转此格点自旋后对系统能量改变的值$dH = H(\{S\}_{n+1}) - H(\{S\}_n)$,

if($dH<0$){// 跃迁几率为1,接受此自旋翻转}else{

// 根据随机数决定是否翻转

if($\xi < \exp(-dH/k_BT)$)// ξ为0到1之间的随机数

自旋翻转

// 否则不翻转

}

} // end for

附录B Ising模型的MPI主函数C程序

```
void main (int argc, char * argv[ ])
{
    FILE   * fid;      // 存储本进程的所测量物理量的输出文件的指针
    char   pname[MPI_MAX_PROCESSOR_NAME];  // 处理器名
    char   filename[3];  // 存储本进程的所测量物理量的输出文件
    int nprocs, namelen, i, j;
    MPI_Init(&argc, &argv);  // MPI初始化
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);  // 取通讯器中进程的数目
    MPI_Comm_rank (MPI_COMM_WORLD, &pid);  // 取当前的进程编号
    MPI_Get_processor_name (pname, &namelen);  // 取当前处理器的名字
        /* 设置处理器(进程)的奇偶性 */
    if (P = =1)
        myparity = 2;  // 进程数为1时
    else if (pid % 2 ! =0)
        myparity = 1;  // 进程数为奇时
    else
        myparity = 0;  // 进程数为偶时
```

```
    /* 近邻近程 ID */
    up_id = (pid + 1)% P;
    lw_id = (pid + P - 1)% P;
    /* 各进程读取数据文件中的参数 */
    readData( );
    // 打开文件,供写入本进程的计算结果
        fid = fopen("dataIn.dat","w"); // dataIn.dat 为输入数据文件
    // 格点自旋组态按均匀随机分布初始化为 spin[i][j] = 1 或 - 1
    for (i = 1; i < = LP; i ++) {
        for (j = 0; j < L; j ++) {
            spin [i][j] = one_rand( );
        }
    }
    // 设置随机数种子
    srand ((unsigned)(time((int *)0) + pid));
    // 调用模拟程序
    simulate (fid);
    MPI_Finalize ( );
}
```

附录 C　Ising 模型的 MPI 子函数 simulate 的 C 程序

```
void simulate(FILE * ofp){
int i,j;
        // 开始时预热系统,扫描整个网格 N 次(N = measure_num)
        for(i = 1; i < = measure_num; i ++){
            // 每隔 interval 次 update 才测量一次(以保持组态的独立性)
            for(j = 1; j < = interval; j ++){
              update(); // 在 update()子函数中将完成对网格的并行化处理,
            }
            // 测量,结果记录到文件中
            measure(ofp);
        }
    }
```

附录 D　Ising 模型的 MPI 子函数 update 的 C 程序

```
    void update(void){
        // 发送底行(第 1 行)给 low_id,再接收数据到顶行
        cache_row(1,lw_id);
        // 行扫描更新
        for (i = LP; i > 1; i - - ){
         // 对当前行执行 Metropolis 抽样
            row_update(i);
        }
        // 先发送顶行数据到 up_id,再接收数据到底端 cache 行
        cache_row(LP, up_id);
            // 更新最底行
        row_update(1);
}
```

附录 E　Ising 模型的 MPI 子函数 measure 的 C 程序

```
void measure(FILE * ofp){
    int i, j, im, ip, jm, jp, this_spin, south_spin, east_spin;
double magnetization = 0. 0;      /* 平均磁场强度 */
double interaction = 0. 0;
/* 扫描格点 LP * L,测量磁场强度和内能 */
for(i = 1; i < = LP; i + + ){ // 每行
im = i - 1;
ip = i + 1;
for(j = 0; j < L; j + + ){ // 每列
jm = (j - 1 + L)% L; // 左行
jp = (j + 1)% L; // 右行
this_spin = spin[i][j];
south_spin = spin[ip][j];
east_spin = spin[i][jp];
magnetization + = this_spin;
interaction + = this_spin * (south_spin + east_spin);
}
}
// 取正值
if(magnetization < 0){
magnetization * = - 1;
}
magnetization = magnetization/(L * L);
interaction = interaction/(L * L);
/* 各进程数据归约 */
MPI_Allreduce ( &magnetization, &totM, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
MPI_Allreduce (&interaction, &totH, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
/* 主进程记录数据到文件 s0 */
if (pid = = 0){
fprintf(ofp, "% 1f % 1f \ n",totM, totH);
fflush(ofp);
}
}
```

Parallel Computing of the Ising Model*

LIU Jun SHEN Yang LUO Xiang-Qian[1)]
(Department of Physics, Zhongshan University, Guangzhou 510275, China)

Abstract We give an introduction to the design and coding of parallel computing for Monte Carlo simulations of lattice systems, by taking the Ising model as an example. The performance results on our PC cluster are also provided. We believe that such information is useful for large scale simulation of lattice QCD.

Key words Monte Carlo simulations, Ising model, lattice gauge theory, parallel algorithm, high performance parallel computers

Received 26 May 2003, Revised 27 July 2003
* Supported by NSFC (10235040), Guangdong Ministry of Education, Foundation of Zhongshan University Advanced Research Center
1) Corresponding author, E-mail: stslxq@zsu.edu.cn